一种基于软件定义网络的服务功能链优化部署机制 *

刘益岑 1，卢　昱 1，王　珊 2，陈兴凯 1，乔文欣 1

(1. 陆军工程大学石家庄校区 装备模拟训练中心，石家庄 050003; 2. 解放军 66172 部队，石家庄 050000)

摘　要： 针对软件定义网络环境下，现有服务链部署方法未能充分考虑全网资源利用率的问题，提出了一种基于高效启发式算法的服务链优化部署机制。首先，给出了服务链部署的总体结构，并引入了整数型线性规划模型对其进行数学建模；其次，提出了一种高效启发式的模型求解算法，该算法以先排序后贪心的方式，能够在满足资源和时延约束下有效利用网络资源和均衡负载。仿真结果表明，与其他部署算法相比，该算法降低了负载均衡度和时间复杂度的同时提高了请求接受率。

关键词： 软件定义网络；服务链；优化部署；整数型线性规划；启发式算法

中图分类号： TP393　　doi: 10.3969/j.issn.1001-3695.2018.03.0226

Dynamic deployment mechanism for service chaining oriented software-defined networking

Liu Yicen[1], Lu Yu[1], Wang Shan[2], Chen Xingkai[1], Qiao Wenxin[1]

(1. *Equipment Training Simulation Center*, *Army Engineering University*, *Shijiazhuang* 050003, *China*; 2. 66712 *Troop*, *Shijiazhuang* 050000, *China*)

Abstract: For the software-defined networking(SDN) environment, the existing service function chaining(SFC) deployment method failed to fully consider the resource utilization of the entire network, and this paper proposed a service chaining optimal deployment mechanism based on the efficient heuristic algorithm. Firstly, this paper gave the overall structure of the service chaining deployment, and introduced an integer linear programming(ILP) model. Secondly, this paper proposed an efficient heuristic algorithm. That algorithm run in "first sort after greed" manner and effectively used network resources and balanced load, under the constraints of resource and delay. Finally, the obtained simulation results show that our proposed algorithm reduces the load balancing and time complexity while improving the request acceptance rate, compared to other deployment algorithms.

Key words: software-defined networking(SDN); service function chaining(SFC); optimal deployment; integer linear programming(ILP); heuristic algorithm

0　引言

随着网络信息量迅猛增长（预计截至 2018 年，全球互联网数据流量将达到 1.6×10^{21} 字节[1]），人们对互联网服务需求不断扩大，传统服务功能静态服务模式带来了诸多问题：一是传统网络中服务功能（如防火墙、入侵检测和网络地址转换等）与专属硬件设备紧耦合，导致其可扩展性差、网络维护成本高等弊端；二是服务功能静态僵化难以适应当前互联网多租户模式的动态需求，造成网络运行的不稳定性。为此，新型动态服务模式研究成为近来研究的热点[2-3]。

目前，软件定义网络[4]（software defined network，SDN）和网络功能虚拟化[5]（network function virtualization，NFV）技术不断发展，SDN 架构采用可编程的逻辑集中控制的方式，可根据不同细粒度需求对服务功能进行编排；NFV 技术改变传统服务功能的嵌入式部署方式，将网络控制功能与硬件设备解耦，为第三方公司开发服务功能提供了条件。两者的结合增强网络的灵活性和可拓展性，达到基础设施资源共享的效果，为动态服务功能链技术的发展提供了支撑。

当前基于软件定义网络的动态服务功能链技术研究还处于起步阶段，在应用中存在很多突出问题，其中服务功能链优化部署是亟待解决的问题之一[6~8]。现有的部署问题研究更多是概念上的验证，在部署策略研究中，现有方法更多是考虑全网资源利用的单一方面，例如文献[9]提出了一种基于禁忌搜索算法的服务功能部署方法，该方法通过设置禁忌表的方式在全局网

收稿日期： 2018-03-20；**修回日期：** 2018-05-07　**基金项目：** 国家自然科学基金资助项目（51377170，61271152）；国家青年科学基金资助项目（61602505）

作者简介： 刘益岑（1990-），男，重庆人，硕士研究生，主要研究方向为 SDN/NFV 相关技术研究（18419764051@163.com）；卢昱（1960-），男，教授，博士，主要研究方向为下一代互联网研究；王珊（1982-），女，高级工程师，博士，主要研究方向为信息网络安全技术；陈兴凯（1988-），男，博士研究生，主要研究方向为可编程网络研究；乔文欣（1992-），女，博士研究生，主要研究方向为虚拟网络的可靠性研究.

络中搜索服务功能最优的部署位置，但该方法只考虑节点资源利用率。文献[10]提出了一种面向 NFV 环境下的服务功能管理机制，该机制主要通过维特比算法在候选节点中选择部署开销最小的服务功能，有效地降低了服务功能的部署成本，但其缺乏对链路部署开销的考虑。文献[11]提出了一种服务功能部署模型，该模型采用 AFM 启发式算法进行求解，但 AFM 仅以物理节点的计算资源容量作为选择策略，未从全局服务路径资源利用考虑。文献[12]通过 Greedy 算法穷举全网满足资源约束及连通性的所有服务路径，并选用链路资源占用量最小的服务路径，但这种方法只侧重于链路的优化选择问题。文献[13]提出了 SDN/NFV 架构下的服务功能部署机制，该机制构建了分层图模型，将时延最小化作为服务路径选择依据，虽然降低了服务功能链的处理时间，但该机制缺少对节点资源的考虑。

上述部署方法从不同的角度进行了探索，其主要研究不足在于：上述部署方法都是单一关注节点资源利用问题，或者单一关注链路资源利用问题，缺乏对全局资源分配方案的研究。因此，本文从全局角度出发，同时关注节点和链路的资源利用率，充分利用虚拟资源灵活分配这一特性，选择合适的功能放置位置和服务路径，以满足资源和时延约束的条件下有效地利用资源、均衡负载。本文首先给出面向 SDN 的服务功能链部署总体结构，并将其建模成整数型线性规划数学模型。其次，针对当前方法未能同时关注节点和链路资源利用率这一难点，设计一种启发式搜索算法求解部署模型。最后，利用负载均衡度、请求接收率和时间复杂度等指标对所提算法评价，并验证所提算法的有效性。

1　网络模型与问题描述

1.1　基于 SDN 的服务功能链部署模型

如图 1 所示，基于 SDN 的服务链部署模型分为编排平面、控制平面和数据平面。编排平面作为虚拟网络功能（virtual network function，VNF）的开发环境，其主要作用是对网络中 VNF 进行集中化管理，并根据不同的业务需求和适用场景构建出 SFC 策略，组合相应的功能组件形成逻辑功能链来管控网络。控制平面根据服务功能资源需求和底层资源信息，并按照一定的部署策略将 VNF 和虚拟链路映射到底层，实现逻辑服务链的部署。数据平面主要包括通用型的硬件设备（如标准化转发设备和 x86 硬件资源设备等），其主要作用是接收控制平面下发的规则，承载实际的服务请求。

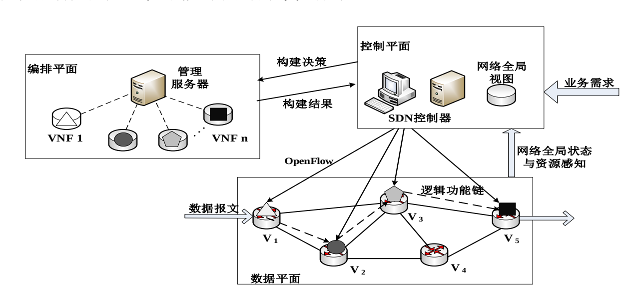


图 1　基于 SDN 的服务功能链部署模型

1.2　服务功能链部署问题描述

服务功能链部署过程可描述为在接收到用户的 SFC 服务请求后，SDN 控制器中的资源管理和 VNF 编排模块根据服务链请求和底层资源状态，按照指定的部署策略找到 VNF 和虚拟链路的优化部署位置，并生成一条满足特定功能、性能的服务路径。为了对基于 SDN 的服务功能链部署问题进行抽象化的表述，可将其归约为两级模型，即 SFC 策略—逻辑功能链和逻辑功能链—具体服务路径，如图 2 所示。SFC 策略—逻辑功能链过程是应用程序通过北向接口实现 VNF 的动态编排组合；逻辑功能链—具体服务路径过程是通过南向接口实现对底层资源的合理分配。

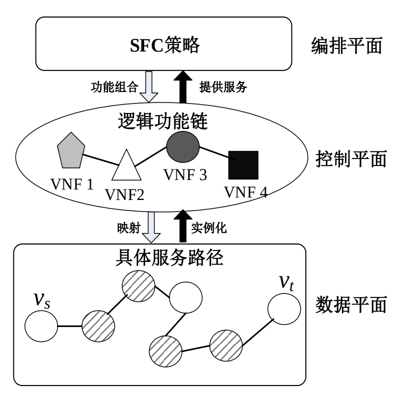


图 2　SFC 策略—逻辑功能链—具体服务路径过程

定义 1　SFC 策略。包含目标集合和服务功能集合，目标集合表示所需执行处理动作的目标，服务功能集合表示数据报文必须依次遍历的服务功能。SFC 策略可以表示为集合 $P=\{v_s,v_t,(c_1,c_2,\ldots c_m)\}$，式中 $C=\{c_1,c_2,\ldots c_m\}$ 表示从源端节点 v_s 到目的端节点 v_t 的数据报文所需经过的服务功能序列，其中 m 表示服务请求的功能数量。

定义 2　逻辑功能链。控制器根据 SFC 策略编排组合 VNF 功能模块形成的逻辑功能链，可用集合 $F=\{f_1,f_2,\ldots f_m\}$ 表示 VNF 逻辑功能链。采用赋权的有向图 G^v 表示全部 VNF 可部署节点及其上下文连接关系，记为 $G^v=(N^v,L^v)$，式中 N^v 表示 VNF 的逻辑节点集合，即 $N^v=\{n_1^v\ldots,n_{i-1}^v,n_i^v,n_{i+1}^v,\ldots n_m^v\mid 1\le i\le m\}$，$L^v$ 表示逻辑节点之间的虚拟链路集合，即 $L^v=\{(n_i^v,n_j^v),\ldots\mid 1\le i<j\le m\}$。

定义 3　具体服务路径。根据指定的目标函数为逻辑功能链中的 VNF 和虚拟链路找到对应的最优放置位置，形成从源端节点到目的端节点的具体服务路径。底层网络由所有物理节点及连接链路组成，可以用赋权无向图 $G^s=(N^s,L^s)$ 表示，其中 N^s 表示底层物理节点集合，即 $N^s=\{n_1^s,\ldots,n_{j-1}^s,n_j^s,n_{j+1}^s\ldots n_k^s\mid 1\le j\le k\}$，$L^s$ 表示物理节点之间的物理链路集合，即 $L^s=\{(n_i^s,n_j^s),\ldots\mid 1\le i<j\le k\}$。

以图 2 的 SFC 策略—逻辑功能链—具体服务路径过程为例来描述从发起服务请求到成功实现业务部署并最终提供相应的功能服务的实例化流程。首先，编排平面根据租户请求，通

过调用北向接口组合形成由4个VNF组成的逻辑功能链；其次，控制平面根据底层网络资源状况，通过南向协议控制设备转发，按照一定的部署策略将逻辑服务链映射到物理底层节点，若逻辑服务功能链映射成功，则分配相应的实例化资源，进而形成一条满足特定功能、性能的服务路径。在具体服务路径中，带有阴影的节点表示部署VNF的物理节点，未带有阴影的物理节点仅仅起到转发数据的作用。

1.3　基于整数规划的优化部署模型

优化部署策略是综合考虑服务资源需求、网络节点资源等情况，按照指定目标函数找到VNF节点和虚拟链路部署的最优位置。本节以占据物理资源最小化为目标，建立整数规划模型（integer liner programming，ILP）。为形式化表述，本文使用的符号如表1所示。

表1　主要的符号定义

符号	含义
$c(n_i^v)$	VNF节点 n_i^v 的计算资源需求量
$r(n_k^s)$	物理节点 n_k^s 的剩余计算资源
$c(n_i^v,n_j^v)$	VNF节点 n_i^v 与 n_j^v 之间虚拟链路的带宽资源需求
$r(n_k^s,n_l^s)$	物理链路 (n_k^s,n_l^s) 的剩余带宽资源
D^P	请求能够容忍最大时延
$D(n_k^s,n_l^s)$	两物理节点之间的最大传输时延
$D(n_k^s)$	功能所在节点 n_k^s 处理数据报文的时延

1）变量

$x_{n_i^v}^{n_k^s}$，二进制变量。在映射过程中，如果物理节点 n_k^s 承载了VNF节点 n_i^v，该变量取值为1，否则取值为0。

$y_{n_i^v,n_j^v}^{n_k^s,n_l^s}$，二进制变量。在映射过程中，如果物理节点 n_k^s、n_l^s 之间物理链路承载了VNF节点 n_i^v、n_j^v 之间虚拟链路，该变量取值为1，否则取值为0。

2）目标函数:

$$\min \sum_{n_i^v\in N^v}\frac{\alpha}{r(n_k^s)+\delta}x_{n_i^v}^{n_k^s}+\sum_{(n_i^v,n_j^v)\in L^v}\frac{\beta}{r(n_k^s,n_l^s)+\delta}y_{n_i^v,n_j^v}^{n_k^s,n_l^s} \quad (1)$$

3）资源约束

$$x_{n_i^v}^{n_k^s}c(n_i^v)\le r(n_k^s),\forall n_i^v\in N^v,n_k^s\in N^s \quad (2)$$

$$y_{n_i^v,n_j^v}^{n_k^s,n_l^s}c(n_i^v,n_j^v)\le r(n_k^s,n_l^s),\\ \forall(n_k^s,n_l^s)\in L^s,(n_i^v,n_j^v)\in L^v \quad (3)$$

4）连接性约束

$$\sum_{(n_i^v,n_j^v)\in L^S,(n_k^s,n_l^s)\in L^v} y_{n_i^v,n_j^v}^{n_k^s,n_l^s}-\sum_{(n_i^v,n_j^v)\in L^S,(n_k^s,n_l^s)\in L^v} y_{n_j^v,n_i^v}^{n_k^s,n_l^s}=\\ x_{n_i^v}^{n_k^s}-x_{n_i^v}^{n_l^s},\forall n_i^v\in N^v,n_k^s\in N^s \quad (4)$$

5）时延约束

$$\sum_{(n_k^s,n_l^s)\in L^S,(n_i^v,n_j^v)\in L^v} y_{n_i^v,n_j^v}^{n_k^s,n_l^s}D(n_k^s,n_l^s)+\sum_{n_i^v\in N^v,n_k^s\in N^S} x_{n_i^v}^{n_k^s}D(n_k^s)\le D^P \quad (5)$$

6）变量约束

$$\sum_{n_i^v\in N^v,n_k^s\in N^S} x_{n_i^v}^{n_k^s}=1,\quad \sum_{(n_i^v,n_j^v)\in L^S,(n_k^s,n_l^s)\in L^v} y_{n_i^v,n_j^v}^{n_k^s,n_l^s}=1 \quad (6)$$

$$x_{n_i^v}^{n_k^s}\in\{0,1\},\ y_{n_i^v,n_j^v}^{n_k^s,n_l^s}\in\{0,1\} \quad (7)$$

式（1）为该部署模型的目标函数，即物理资源占用量最少，以实现服务功能链部署成本最小化，式中 α 和 β 为两个缩放参数，用于调整VNF映射和链路映射成本的影响因子。δ 表示极小值，用于保证分母的非零性。为了提高底层物理资源的利用率，目标函数中通过将 $r(n_k^s)$ 和 $r(n_k^s,n_l^s)$ 作为分母使得VNF映射和链路映射策略更倾向于物理资源剩余多的节点或链路。

式（2）（3）给出了计算资源和带宽资源约束条件。在计算资源约束方面，表示底层物理节点剩余资源应满足VNF实例化所需的计算资源需求。在带宽资源约束方面，底层物理链路剩余资源应满足请求中链路带宽资源需求。

式（4）为连接性约束。若在VNF节点映射阶段，VNF节点 n_i^v 和 n_j^v 分别映射到底层节点 n_k^s 和 n_l^s，那么在逻辑链路映射阶段，逻辑链路 (n_i^v,n_j^v) 将被映射到从节点 n_k^s 到节点 n_l^s 的一条底层路径上。在源节点 n_k^s 上，流出的流量为1，流入的流量为0，所以 $\sum_{(n_i^v,n_j^v)\in L^S,(n_k^s,n_l^s)\in L^v} y_{n_i^v,n_j^v}^{n_k^s,n_l^s}-\sum_{(n_i^v,n_j^v)\in L^S,(n_k^s,n_l^s)\in L^v} y_{n_j^v,n_i^v}^{n_k^s,n_l^s}=1$；在源节点 n_l^s 上，流出的流量为0，流入的流量为1，所以 $\sum_{(n_i^v,n_j^v)\in L^S,(n_k^s,n_l^s)\in L^v} y_{n_i^v,n_j^v}^{n_k^s,n_l^s}-\sum_{(n_i^v,n_j^v)\in L^S,(n_k^s,n_l^s)\in L^v} y_{n_j^v,n_i^v}^{n_k^s,n_l^s}=-1$；而在其它节点上，流入流出的流量均为1，所以 $\sum_{(n_i^v,n_j^v)\in L^S,(n_k^s,n_l^s)\in L^v} y_{n_i^v,n_j^v}^{n_k^s,n_l^s}-\sum_{(n_i^v,n_j^v)\in L^S,(n_k^s,n_l^s)\in L^v} y_{n_j^v,n_i^v}^{n_k^s,n_l^s}=0$。

式（5）为服务路径的时延约束。服务提供商为租户成功将服务请求映射到底层物理网络过程中，构建出的一条端到端服务路径，时延表示数据流从源端节点发出到目的节点所耗费的时间，时延约束确保满足网络的服务等级协定（SLA）。

式（6）为逻辑功能链的映射约束，确保同一个服务请求中的服务只能映射到底层唯一的节点上。式（7）为 $x_{n_i^v}^{n_k^s}$ 变量和 $y_{n_i^v,n_j^v}^{n_k^s,n_l^s}$ 变量的二进制约束。

2 算法描述

针对上文提出的服务功能链部署模型，可利用优化理论将其归约为虚拟网络映射问题[14~16]。虚拟网络映射问题已被证明是 NP-Hard 问题[17]，因此服务功能链部署问题同样为 NP-Hard 问题，即在 $NP \neq P$ 条件下，不可能存在多项式时间算法来解决该问题，通常采用高效的搜索算法以求得问题的近似解。传统虚拟网络映射过程中的两步映射算法[18]将映射过程分为节点映射和链路映射，但服务链部署过程中的逻辑功能链映射和传统虚拟网络映射不同的是，需要综合考虑服务链的端系统、实例化资源分配和编排顺序等特性。本文根据服务功能链部署的特性，设计一种“先排序后贪心”的启发式搜索算法，以快速且高效地求解服务功能链部署模型。

2.1 基于高效启发式算法的服务链部署

当前对服务功能链部署问题更多是单一关注节点资源利用问题，或者单一关注链路资源利用问题。针对这一难点，本文提出了一种高效启发式搜索算法（greedy node embeding with k-shortest path link embeding algorithm，G-kSP），该算法综合考虑租户的服务资源需求、底层物理剩余资源和 QoS 等情况，在服务功能链部署过程中同时关注 VNF 放置和路径选择两阶段，从而得到全局最优的资源分配策略。

为了对逻辑功能链中 VNF 资源需求情况和底层物理节点资源剩余情况更为精确化的描述，本节在此给出 VNF 综合资源需求函数 $C(n_i^v)$ 和物理节点综合剩余资源函数 $R(n_k^s)$ 的概念。其定义分别如下：

$$C(n_i^v) = c(n_i^v) \sum_{e_i^v \in E(n_i^v)} c(e_i^v) \tag{8}$$

其中：$E(n_i^v)$ 表示 VNF 节点 n_i^v 所有邻接虚拟链路集合，$c(n_i^v)$ 表示 VNF 节点 n_i^v 实例化所需的计算资源量，$r(e_i^v)$ 表示虚拟链路实例化所需的带宽资源。

$$R(n_k^s) = r(n_k^s) \sum_{e_k^s \in E(n_k^s)} r(e_k^s) \tag{9}$$

其中：$E(n_k^s)$ 表示底层物理节点 n_k^s 所有邻接物理链路集合，$r(n_k^s)$ 表示物理节点 n_k^s 当前剩余的计算资源，$r(e_k^s)$ 表示物理链路当前剩余的带宽资源。

为了简化传统启发式算法的搜索空间，提高求解全局资源分配策略的效率，针对式（1）~（7）所示的部署模型，本文设计的 G-kSP 算法是一种先排序后贪心高效搜索方法，其中采用排序的方式旨在简化算法的搜索空间，在不进行全局搜索的情况下就能得到最优服务路径，而在贪心选择过程中，将优先考虑资源剩余量最多的物理节点或链路作为贪心策略，贪心策略对每一子问题操作产生直接影响，得到每个子问题最优的资源分配策略，进而快速、高效地逼近模型目标函数。同时，在迭代过程中算法采用回溯的机制，即当前迭代的 VNF 或虚拟链路搜索不到满足资源约束条件的解时，应当回溯到上一阶段迭代可映射集合中的次优解。G-kSP 算法的运行过程具体描述如表 2 所示。

表 2 G-kSP 算法具体过程

输入： SFC 请求 $P = \{v_s, v_t, (c_1, c_2, \ldots c_m), \tau\}$，底层物理网络 $G^s = (N^s, L^s)$。

输出： VNF 映射集合 $\{M_N(n_i^s) | \forall n_i^s \in N^s\}$、虚拟链路映射集合 $\{M_L(n_i^s, n_j^s) | \forall (n_i^s, n_j^s) \in L^s\}$。

a) 构建 VNF 队列 Q。根据在线到达的请求构建 SFC 策略，进而编排组合 VNF 形成具有链式结构的虚拟网络层 G^v，计算集合 N^v 中每个 VNF 节点的资源需求量 $C(n_i^v)$，按照 $C(n_i^v)$ 由大到小进行重新排列，并将其放入队列 Q 中。

b) 构建可映射节点集合 M。根据底层物理节点 $r(n_k^s)$ 及 $r(e_k^s)$ 资源信息，计算可映射物理节点 n_k^s 的综合剩余资源量 $R(n_k^s)$，并利用式（2）~（6）判断该物理节点的剩余资源是否满足资源约束，将满足约束条件的底层物理节点形成可映射集合 M。

c) 选择 VNF 节点最优位置 $M_N(n_i^v)$。取出队列 Q 中第一个需要部署的 VNF，利用式（10）搜索可映射集合 N^s 中的物理节点，将 VNF 关联到物理节点 n_k^s 上，若成功将 VNF 映射到物理节点 n_k^s 上，则 $x_{n_i^v}^{n_k^s} = 1$，并记录到集合 M_N 中；否则，回溯到上一步迭代可映射集合 M 中的次优解。更新队列 Q。

$$n_k^s = \left\{ \hat{n}_k^s \middle| \hat{n}_k^s = \arg\max R(n_k^s), \forall n_k^s \in N^s \right\} \tag{10}$$

d) 得到 VNF 映射集合 M_N。判断队列 Q 中是否还有 VNF 没有完成映射，若是则转到步骤 2，执行队列 Q 中下一个 VNF 节点；否则，完成 VNF 映射，得到集合 M_N。

e) 构建路径集合 R。完成 VNF 映射后，根据集合 M_N，按照服务功能链顺序 φ_P，以链路剩余带宽 $r(n_k^s, n_l^s)$ 为权重，利用 *K-Dijkstra* 算法计算从源端节点 v_s 到目的端节点 v_t 的 k 条最短路径集合 R，记为 $R = \{r_1, r_2, \ldots, r_k\}$。

f) 选择虚拟链路最优位置 $M_L(n_i^s, n_j^s)$。将集合 R 按照带宽资源占用量 $c(n_i^v, n_j^v)$ 进行由小到大的排序，并选取第一条路径，则 $y_{n_i^v, n_j^v}^{n_k^s, n_l^s} = 1$，记录到集合 M_L 中；否则，回溯到最短路径集合 R 中的次优解。

g) 服务时间监控和资源分配。完成虚拟链路映射后，利用式（7）判断是否满足服务功能链 QoS 指标，若满足则底层资源池为其分配相应实例化资源，以完成 SFC 部署过程，并更新底层资源状态；否则，收回为其分配的资源，服务功能链部署失败。

2.2 算法复杂性理论分析

在服务功能链部署过程中，假设存在 n 个 VNF 需要映射，

底层网络中 m 个物理节点和 M 条物理链路。G-kSP 算法在 VNF 映射过程中，首先将 n 个 VNF 根据所需资源按照从大到小顺序进行排列，VNF 映射过程的计算量主要在于资源需求量排序和底层剩余资源量的搜索，其复杂度为 $O(n\log n)$，而在虚拟链路映射过程中，由于每条虚拟链路映射到底层物理网络中采用 *K-Dijkstra* 算法，其时间复杂度为 $O(K+M)$。综上所述，G-kSP 算法的时间复杂度为 $O(K+n\log n+M)$，即可等价于 $O(n\log n)$。

3 实验评估与分析

为了验证本文所提出的部署算法（记为 G-kSP）的有效性，本文选取其它两种典型的部署算法进行对比（如表 3 所示），选取负载均衡度，请求接受率和时间复杂度这三种重要部署评价指标来验证 G-kSP 算法的有效性，其中负载均衡度反映算法搜索的均衡程度，请求接受率反映算法在解空间的搜索性能，而请求处理时间反映算法在计算性能。

表 3 比较算法

算法	描述
G-kSP	本文所提出的基于全局资源利用的高效启发式搜索算法
TS	基于文献[9]所提出的通过设置禁忌表的方式在全局网络中搜索服务最优的部署位置的算法，这是一种侧重于节点资源优化利用的方法
Greedy	基于文献[12]所提出的通过赋予考虑低时延链路较高优先级的算法，侧重链路资源的优化利用

3.1 实验环境和参数设置

本实验在配置为 Intel Core i7-3770 3.60 GHz、8 GB 内存的 Linux 系统 PC 机上运行。网络拓扑结构是利用 GT-ITM[19] 工具生成，算法程序通过 Matlab 软件运行。与文献[20]类似，本文采用包含 6 个相同物理节点的网络拓扑结构的测试例子作为底层基础设施来进行实验，与一个轻量级的云数据中心网络环境类似。如图 3 所示，网络拓扑由 6 个节点和 14 条链路组成（节点上的数字代表节点标签号），以节点 1 为数据流量入口，节点 6 作为流量的出口。假设所有节点部署在云数据中心，并连接所有节点能够承载服务功能，物理节点资源和链路带宽剩余资源容量服从[1000,1500]的随机分布，链路和节点的单位映射成本均为 1。每条 SFC 请求由不同类型的服务功能组成，其数量服从[2,5]的随机分布，每条 SFC 节点和链路的资源需求量均服从[0.5,1]的均匀分布，底层网络支持的服务类型数量设为 10，每个节点随机提供其中的 1-5 种。

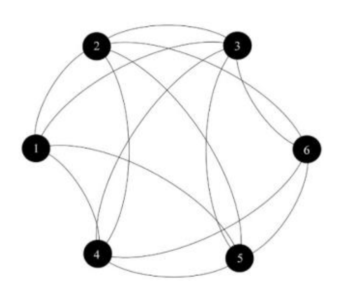


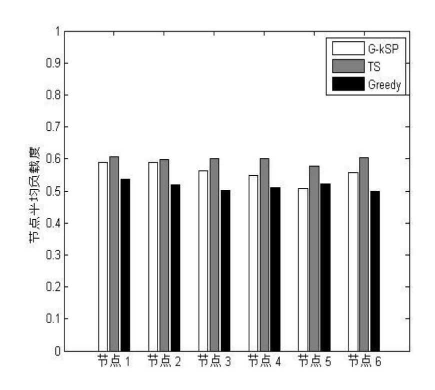
图 3 仿真实验网络拓扑

3.2 算法性能对比

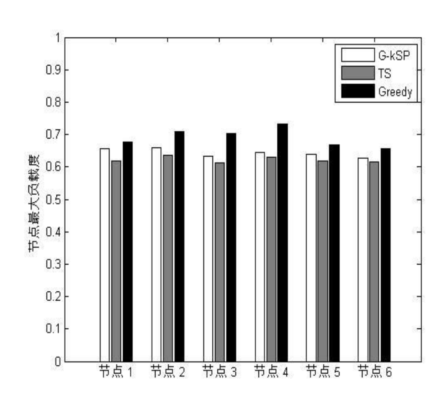
a)节点负载均衡度,表示物理节点的负载积累情况，反映出算法在节点资源利用率上的优劣程度，其取值越小越好，各物理节点上负载越均衡越好。记物理节点的负载均衡度计算式为

$$LB_{n_i^s}=\sum_{n_k^s\in M_N,n_i^v\in N^v}\frac{c(n_i^v)}{r(n_k^s)} \tag{11}$$

实验中生成 $10\sim100$ 次 SFC 请求强度，分别统计 3 种算法的节点负载均衡度，如图 4 所示。



(a)节点平均负载均衡度比较



(b) 节点最大负载均衡度比较

图 4 节点负载均衡度性能比较

由图 4 可知，基于 TS 的部署方法能在获得最大平均负载均衡度的同时，节点最大负载均衡度更小，表明该算法在均衡节点负载度的效果更好，反之 Greedy 算法的效果较差，G-kSP 算法居中。原因在于 TS 算法以底层节点的计算资源容量作为选择策略，在反复迭代过程中能搜索到较好的节点选择方案，使得节点的资源能够得到有效利用，而 Greedy 算法优先考虑低时延链路上的节点，容易造成部分功能节点负载均衡效果不佳。

b)具体路径负载均衡度,表示整条服务功能链的负载累积情况。反映出算法在全网资源利用率上的优劣程度，其取值越小越好。记具体执行路径的负载均衡度计算式为

$$LB_{RSP}=\omega_N\sum_{n_k^s\in M_N,n_i^v\in N^v}\frac{c(n_i^v)}{r(n_k^s)}+\omega_L\sum_{(n_k^s,n_l^s)\in M_L,(n_i^v,n_j^v)\in L^v}\frac{c(n_i^v,n_j^v)}{r(n_k^s,n_l^s)} \tag{12}$$

式中缩放因子 ω_N 和 ω_L 用于调整节点负载和链路负载的侧重程度。

实验中生成10～100次 SFC 请求强度，分别统计 3 种算法的服务路径负载均衡度，如图 5 所示。由图 5 可知，随着请求到达强度的增加，可用的资源量减少，曲线逐渐趋近平缓。随着资源瓶颈的出现，请求接受率逐渐降低，服务路径构建失败的次数增多。随着请求到达强度增加，基于 G-kSP 算法的部署方法最能有效平衡服务路径的负载，由于 TS 算法以及 Greedy 算法均未能从全局最优路径的角度进行资源分配，从而致使网络服务路径的负载相对不平衡，但 Greedy 算法赋予考虑低时延链路较高的部署优先级，容易导致部分链路频繁复用，由此过早陷入资源瓶颈，因此 TS 算法的服务路径负载均衡性能上优于 Greedy 算法。

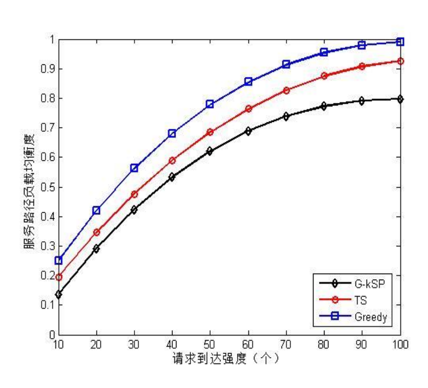


图 5　具体路径负载均衡度比较

c)请求接受率,表示满足资源、时延等约束的请求接受比例，其反映算法的在解空间中的搜索性能，取值越大越好。请求接受率可表示为成功部署到底层物理网络的SFC条数与SFC请求总条数之比，记请求接受率 η 的计算公式为

$$\eta = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} SFC_{accepted}}{\sum_{t=0}^{T} SFC_{total}} \tag{13}$$

其中 $\sum_{t=0}^{T} SFC_{accepted}$ 表示从时刻 $t=0$ 到时刻 T 成功部署到底层物理网络的 SFC 条数；$\sum_{t=0}^{T} SFC_{total}$ 表示到达的 SFC 请求总条数。

实验中生成10～100次 SFC 请求强度，分别统计 3 种部署算法的请求接受率，如图 6 所示。由图 6 可知，在相同条件下，基于 G-kSP 算法的平均请求接受率约为 95.4%，TS 部署算法平均请求接受率约为 92.3%， Greedy 部署算法平均请求接受率约为 81.7%。分析其原因是 G-kSP 算法主要从全局最优路径考虑，降低了整条服务功能链的负载均衡程度，提高了底层物理资源的利用率，占据更少的资源空间，从而尽最大限度为队列中其他请求完成部署留出实例化资源，增加了请求接受率；而 Greedy 算法主要考虑链路资源利用，增加了节点处理的排队时间和底层网络负载，从而降低了请求接受率。而 TS 算法主要考虑了节点资源情况，能够通过反复迭代计算搜索最优的部署方案，相比于 Greedy 算法改善了服务请求接受率。

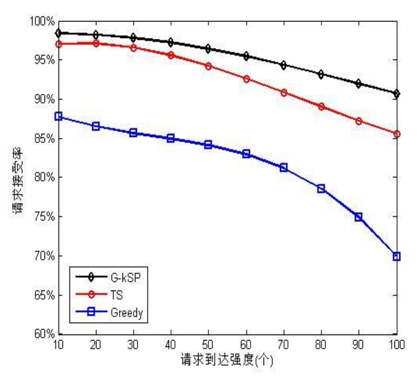


图 6　请求接受率比较

d）时间复杂度,表示从发出请求到完成部署过程的运行时间，时间复杂度反映算法在计算性能方面的优劣程度，运行时间的取值越小越好。实验中生成10～100次 SFC 请求强度，分别统计 3 种算法的请求处理时延，如图 7 所示。

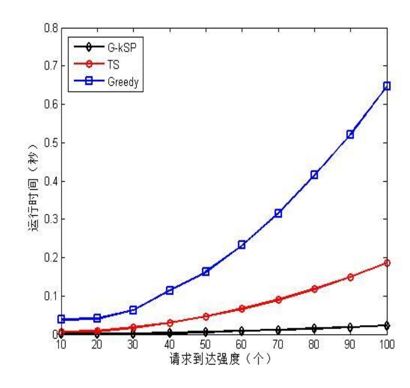


图 7　算法时间复杂度比较

由图 7 可知，随着请求到达强度增加，G-kSP 算法的处理时延较小，能够更快地完成服务功能链的部署，反之 Greedy 算法的处理时延较大，其时间复杂度约为 G-kSP 算法的 10~12 倍，TS 算法居中。分析其原因是 Greedy 算法实质是一种两步映射算法，即分为 VNF 节点映射和逻辑链路映射，在 VNF 映射阶段过程中穷举出底层满足约束条件的节点，时间复杂度为 $O(m^n)$，在逻辑链路映射过程中通过最短路径选择节点之间带宽占用最少的路径，时间复杂度为 $O(m^2)$，完成映射的 VNF 节点需要等待队列中下一个 VNF 节点完成映射后才能建立服务路径，因此 Greedy 算法不仅搜索空间度大，而且虚拟链路映射容易受到 VNF 映射的影响，导致队列中 VNF 等待的时间较长，算法运行效率低，则总的时间复杂度则为 $O(m^n+m^2)$。而 TS 算法的处理时延受禁忌表长度 λ 的影响，随着 λ 增加会造成算法存储空间扩大，计算处理时延增加，TS 算法的时间复杂度可表示为 $O(m^2+m\lambda)$。结合 3.2 节中 G-kSP 算法复杂度的理论分析，图 7 中各曲线的变化趋势与理论分析基本一致。

4　结束语

本文提出的服务功能链部署方法能够同时关注部署过程中节点和链路的资源利用率问题。首先，针对服务功能链优化部

署问题，给出面向 SDN 的服务功能链部署总体结构，将部署总体结构归约为 SFC 策略—逻辑功能链—具体服务路径两级模型，并建模成整型线性规划数学模型。此外，针对当前部署方法中未能完全考虑全局资源利用这一难点，设计了一种先排序后贪心的启发式搜索算法，与禁忌部署算法和贪婪启发式算法对比，本文方法在负载均衡度、请求接收率和时间复杂度等性能指标上有良好的性能，并能够更快速、更有效地得到最优资源分配方案。目前，业界对基于 SDN 的服务功能链研究不断深入，下一步将针对可靠性感知下的服务功能链部署研究，以满足网络服务链可靠性需求和电信级的网络服务质量。

参考文献：

[1] Cisco. Cisco visual networking index: forecast and methodology [R/OL]. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360. html

[2] Bhamare D, Jain R, Samaka M, *et al.* A survey on service function chaining [J]. Journal of Network & Computer Applications, 2016, 75 (C): 138-155.

[3] Medhat A M, Taleb T, Elmangoush A, *et al.* Service function chaining in next generation networks: state of the art and research challenges [J]. IEEE Communications Magazine, 2017, 55 (2): 216-223.

[4] Mckeown N, Anderson T, Balakrishnan H, *et al.* OpenFlow: enabling innovation in campus networks [J]. ACM SIGCOMM Computer Communication Review, 2008, 38 (2): 69-74.

[5] ETSI. Network functions virtualization [R]. NFV White Paper, 2012.

[6] Takacs A, Green H, Shirazipour M, *et al.* Network function placement for NFV chaining in packet//optical datacenters [J]. Journal of Lightwave Technology, 2015, 33 (8): 1565-1570.

[7] Huang Meitian, Liang Weifa, Xu Zichuan, *et al.* Throughput maximization in software-defined networks with consolidated middleboxes [C]// Local Computer Networks. 2016: 298-306.

[8] Kim S, Park S, Kim Y, *et al.* VNF-EQ: dynamic placement of virtual network functions for energy efficiency and QoS guarantee in NFV [J]. Cluster Computing, 2017, 20 (3): 2107-2117.

[9] Mijumbi R, Serrat J, Gorricho J L, *et al.* Design and evaluation of algorithms for mapping and scheduling of virtual network functions [C]// Network Softwarization. 2015: 1-9.

[10] Bari M F, Chowdhury S R, Ahmed R, *et al.* On orchestrating virtual network functions [C]// Proc of International Conference on Network and Service Management. 2015: 50-56.

[11] Zeng Menglu, Fang Wenjian, Zhu Zuqing. Orchestrating tree-type vnf forwarding graphs in inter-dc elastic optical networks [J]. Journal of Lightwave Technology, 2016, 34 (14): 3330-3341.

[12] Lukovszki T, Rost M, Schmid S. It's a match!: near-optimal and incremental middlebox deployment [J]. ACM SIGCOMM Computer Communication Review, 2016, 46 (1): 30-36.

[13] Dwaraki A, Wolf T. Adaptive service-chain routing for virtual network functions in software-defined networks [C]// Proc of Workshop on Hot Topics in Middleboxes and Network Function Virtualization. New York: ACM Press, 2016: 32-37.

[14] Even G, Rost M, Schmid S. An approximation algorithm for path computation and function placement in SDNs [C]// Proc of International Colloquium on Structural Information and Communication Complexity. Cham: Springer, 2016: 374-390.

[15] Schmid S. Online Admission control and embedding of service chains [C]// Proc of International Colloquium on Structural Information and Communication Complexity. New York: Springer-Verlag, 2015: 104-118.

[16] Moens H, Turck F D. VNF-P: a model for efficient placement of virtualized network functions [C]// Proc of International Conference on Network and Service Management. 2014: 418-423.

[17] Chowdhury M, Rahman M R, Boutaba R. ViNEYard: virtual network embedding algorithms with coordinated node and link mapping [J]. IEEE//ACM Trans on Networking, 2012, 20 (1): 206-219.

[18] Yu Minlan, Yi Yung, Rexford J, *et al.* Rethinking virtual network embedding: substrate support for path splitting and migration [J]. ACM SIGCOMM Computer Communication Review, 2008, 38 (2): 17-29.

[19] Calvert K L, Bhattacharjee S. How to Model an Internetwork [C]// Proc of IEEE INFOCOM. 1996: 594.

[20] Sahhaf S, Tavernier W, Rost M, *et al.* Network service chaining with optimized network function embedding supporting service decompositions [J]. International Journal of Computer & Telecommunications Networking, 2015, 93 (P3): 492-505.